HyperCardとTurtleGraphics

町　田　規　雄

1. はじめに

HyperCardに「TurtleGraphics」の機能を盛り込み，Logoとほぼ同じような，図形を描くProgramを対話的に開発する環境を作った．以下はその方法の概要である．

2. HyperCardの概要

HyperCardは1987年にMacPaintの作者Bill Atkinsonによって作られた絵と文章とProgramを一元的に管理する能力をもったMacintoshのためのソフトウェアで，いろいろなアプリケーションを作るための「道具箱　(テキスト編集，計算，データベース処理，グラフィック・デザインのための道具が入ってる)」とみなすことができる．それらの道具をProgram言語HyperTalkで組み合わせることによりMacのすぐれたインターフェースをいかしたアプリケーション(HypecardではStackという)が簡単に作ることができ，しかもユーザーは他の言語で作成した外部CommandをHyperTalkに「差し込んで」機能を拡張できる．(ない道具は自分で補充できる)

Stackの例として，Quatations (HyperCard1.0と共にApple社より提供されたStack)を示しながらHyperCardの概要を説明する．

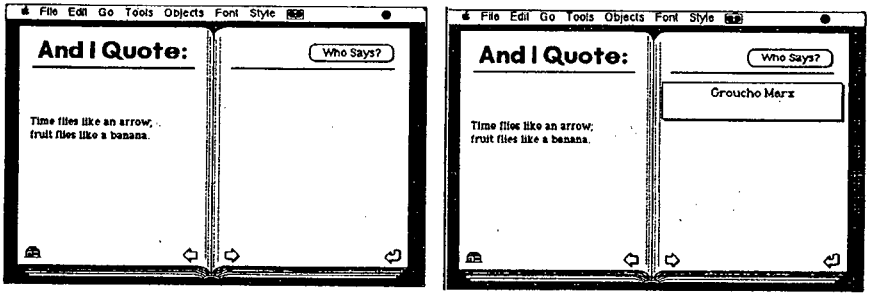


図2-1

このStackは左側に書かれた言葉を誰が言ったかを示す．最初は答えが隠されているが，Who Says?と書かれたButtonをマウスでクリックすると，右側に答えが示される．

このようにHyperCardのStackには，少なくとも１枚のCardがあり，CardにはButton(ユーザーとのやり取りを簡便にするための手段，これをマウスでクリックすることにより命令を実行する)・Field(Text・数値を保存表示をするための枠) ・Picture(絵)をおくことができる．

また共通の形式のCardをつくるために，複数のCardで共通のBackgroundをもつことができる．図2-2にQuatations のBackgroundとそれに含まれるButton,Fieldを示す．

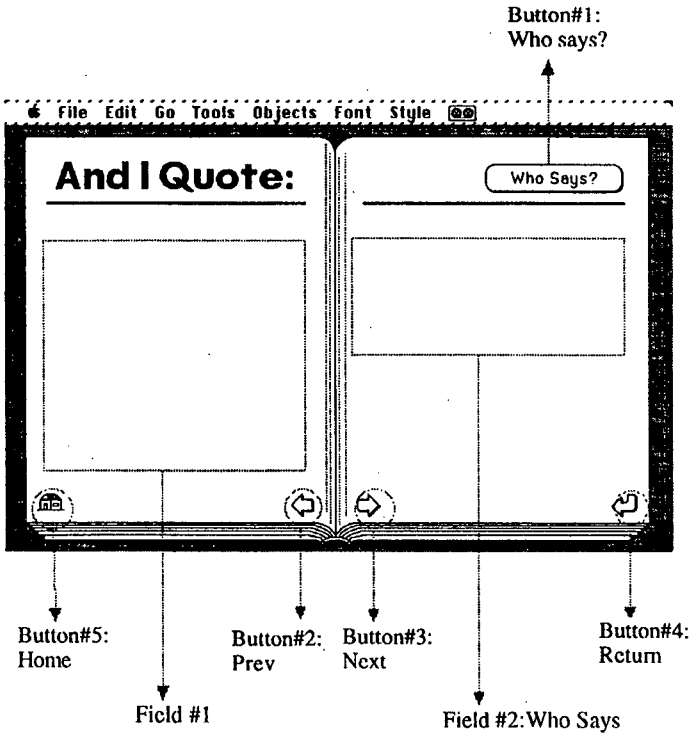


図2-2

このようなStackを動かすためにはシステム環境設定などを行うHomeStack(図2-3)が必要である．またこのStackは他のStackのIndexとしての役割もしている．

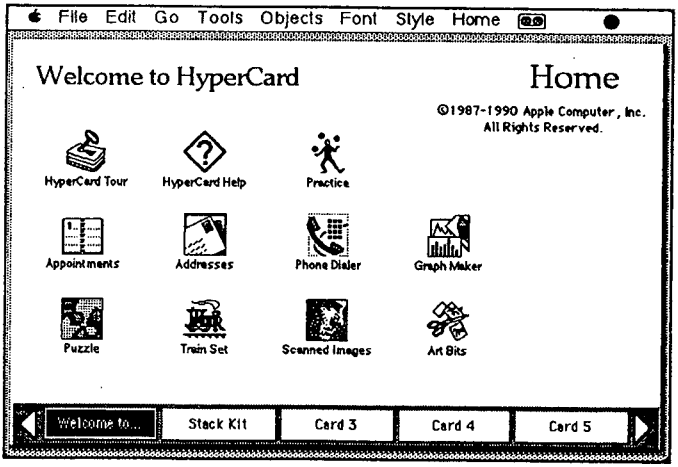


図2-3

HyperCardではこれらButton,Field,Card,Background Card, Stack, HomeStack, HyperCardをObjectと呼ぶ.

HyperCardは緩やかなObject指向環境であり．HyperTalkはObjectに対してMessageを送るという方法で命令を伝え実行していく.

Messageにはシステムが出すもの(ユーザーがマウスをクリックしたとき発生するmouseUpや，Stackを開けたときに発生するopenStackなど)とユーザーが記述し，対応する動作を適切なObjectに書き込んでおくものとがある.

Objectには送られてきたMessageを受けとめるMessage handlerを書き．Messageを受けとめたときの処理を定義する.

handlerは

on <message> [parameter]

<command 1>

<command 2>

.....

<command n>

end <message>

の形をとる．command 1からcommand nまでのScriptによりMessageを受けてからの処理を記述する.

このようにHyperCardのすべてのObjectは，自分に関わるHyperTalkのScriptを持つこと

ができる．

Mcssageが発生すると，図2-4のようにMessageは上から下に水のように流れ，このMessageを受け取るMessage handlerまで落ち処理が実行される．

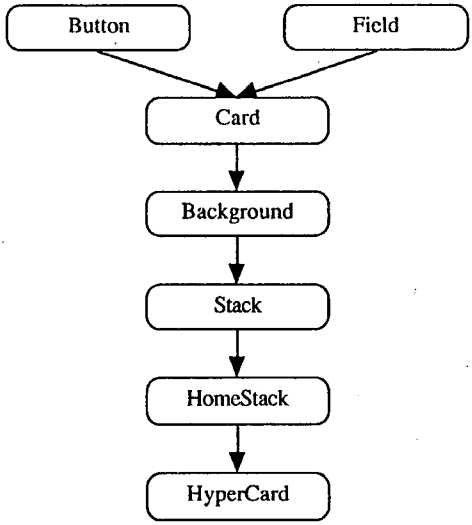


図2-4

QuatationsのStackに含まれるScriptを以下に，示す．

```
SCRIPTS FOR STACK: Quotations
=====================================================
** STACK SCRIPT ************************************
on openStack
  push recent card
  hide message box
end openStack
```

->このStackを開いたとき，Stackを開く前のCardを覚えておく．Message Boxが開いていたらそれを閉じる．

```
on mouseUp
  hide background Field "Who Says"
end mouseUp
```

->マウスが押されたらField#2:Who Saysを隠す．

```
** BACKGROUND #1 ***********************************
on closecard
```

```
  hide Field 2
end closecard
```
->Cardを閉じるときField#2を隠す.

```
** BKGND #1, Field #2: Who Says ***********************************
on mouseUp
  hide Field "Who Says"
end mouseUp
```
->Field#2をMauseで押したときはField#2を隠す.

```
** BKGND #1, Button #1: Who Says? ***********************************
on mouseUp
  set visible of Field "Who Says" to not the visible of Field "Who Says"
end mouseUp
```
->Who Says? ButtonをMauseで押したときはField#2を見えるようにする.

```
** BKGND #1, Button #2: Prev ***********************************
on mouseUp
  visual barn door open
  go to prev card
end mouseUp
```
-> 視覚効果barn door(画面の中央が割れ，観音開きのドアを開けるようにして変化する)してから，前のCardに移る.

```
** BKGND #1, Button #3: Next ***********************************
on mouseUp
  visual barn door close
  go to next card
end mouseUp
```
-> 視覚効果barn doorしてから，次のCardに移る.

```
** BKGND #1, Button #4: Return ***********************************
on mouseUp
  visual effect iris close
  pop card
end mouseUp
```
-> 視覚効果iris close(カメラの絞りを絞り込むような変化）してから，このStackに入る前に覚えておいたCardに戻る.

```
** BKGND #1, Button #5: Home ***********************************
on mouseUp
  go home
end mouseUp
```
-> Home Stackに戻る.

このように，HyperCardのProgramはBasic,Pascalなどの言語と大きく異なり，Programの最初から最後までを制御するようなものは一つもなく，Programは個々のObjectの属性として定義される．つまり，ProgramはButton，Fieldがおされたとき，HyperCardがどのように行動すればよいかを書いたものである．そのための小さなScriptを作るのが，HyperCardの特徴である．

MacintoshのOSには，複数のProgramの間で共通の部品として扱えるようにData(字体の情報，Windowの形式，Programの自身など)を標準の形式で扱う，Resourceと言う概念がある．他の言語で作成した，HyperTalkのCommand,Functionはそれぞれ XCMD,XFCNと呼ばれるが．それらは，Resourceとして作成し，Stack Resource，Home Resourceに書き込まれる．この機能によりHyperTalkを拡張することができる．Resourceを加えるとMessageの伝わる様子は図2-5のようになる．

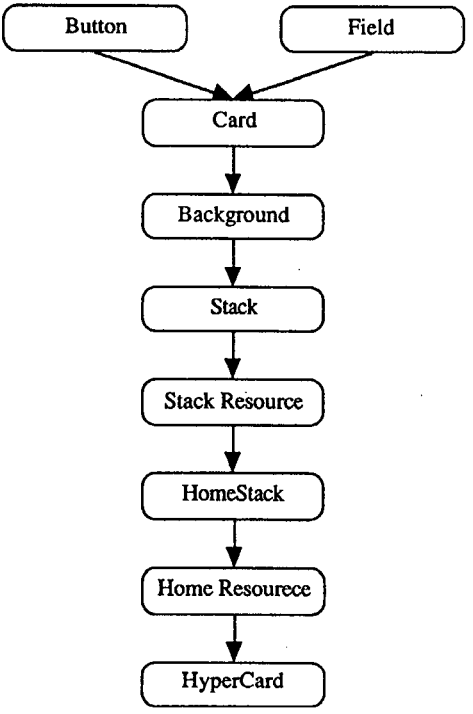


図2-5

3. TurtleGraphics

開発した「TurtleGraphics」（以下これをHyperTurtleと呼ぶ)は，S.PapertとMITにおけるLogoの開発者たちによって考案された「Turtle」という概念に基づいたものでる．1匹のTurtleを画面上で移動させ，その軌跡によって図形を描く．Turtleはペンを持っていて，このペンを「下げた」状態でTurtleが移動すると，Turtleが通過した後に，ペンによる図形が描かれる．一方，ペンを「上げた」状態でTurtleが移動しても，何も描かれないというもので，Turtleを適切に移動することによって，複雑な図形を容易に描くことが出来る．

このHyperTurleの座標は，画面の中心(0，0)を中心とするx-y座標系を採用する，Turtleはこの座標平面の任意の位置に移動できるが，グラフィック画面には，$-256 \leqq x \leqq 256$ $-165 \leqq y \leqq 160$だけが常に表示される．HyperTurtleではTurtleは表示されない．

外部Commandして拡張したHyperTurtleのCommand（XCMD）と関数(XFCN)を表－1に示す．

名前	種別	用法	機能
Forward	XCMD	Forward <数値>	今の向きに<数値>だけ進む．
Back	XCMD	Back <数値>	今の向きと逆向きに<数値>だけ進む．
Right	XCMD	Right <数値>	向きを右に<数値>だけ回転させる．
Left	XCMD	Left <数値>	向きを左に<数値>だけ回転させる．
PenUp	XCMD	PenUp	移動により線を引かないようにする．
PenDown	XCMD	PenDown	移動により線が描かれる．
ClearScreen	XCMD	ClearScreen	画面を消す．
SetXY	XCMD	SetXY <数値1> <数値2>	座標（<数値1>,<数値2>）に移動する．
SetX	XCMD	SetX <数値>	現在位置のx座標を<数値>に変更し，その位置に移動する．
SetY	XCMD	SetY <数値>	現在位置のy座標を<数値>に変更し，その位置に移動する．
Heading	XFNC	Heading()	現在の向きを返す関数．
SetHeading	XCMD	SetHeading <数値>	向きを<数値>に設定する．

表－1

注意：これらのCommandを使うにはあらかじめラインツールを選択していなければならない．XCMDのルーチン内でもラインツールを選択出来るが速度を少しでも速くするためである．

これらをTHINK Pascal3.0, Wild Thingsに含まれるHyperTalkのCallback Routine, GlueRoutine, GlueRoutine Utility を用いて作成した.

Turtle Graphics の機能を盛り込むためには, 現在のTurtleの位置, 向き, Penを下げた状態か上げた状態かをglobal変数として管理する必要がある.
それを, 管理するのがUNIT globalであるが, これを List-1に示す.

List-1 global.tlsp

```
{ global variables for TURTLE GRAPHICS ( HYPER CARD ) }
unit global;
interface
  uses
     HyperXCMD, XCMDGlue;
  type
     global = record
              x: Extended;{TurtleのX座標}
              y: Extended;{TurtleのY座標}
              v: Extended;{Turtleの向き}
              isPendown: boolean;{ペンの状態}
          end;
  const
     CenterX = 256;{画面の中心のX座標}
     CenterY = 175;{画面の中心のY座標}
  {HyperCardより, global変数をよみとる手続き}
  procedure GetGlobalValue (paramPtr: XCmdPtr;var gl:global);
  {HyperCardのglobal変数を書き換える手続き}
  procedure SetGlobalValue (paramPtr: XCmdPtr;gl:global);

implementation
{procedure GetGlobalValue (paramPtr: XCmdPtr,var gl: global) }
  procedure GetGlobalValue;
     var
        MYTIMEHAND: handle;
        Str: str255;
  begin
     MYTIMEHAND := GetGlobal(paramPtr, 'x');
     hlock(MYTIMEHAND);
```

```
    ZeroToPas(paramPtr, MYTIMEHAND^, Str);
    gl.x := StrToExt(paramPtr, Str);
    DisposHandle(MYTIMEHAND);

    MYTIMEHAND := GetGlobal(paramPtr, 'y');
    hlock(MYTIMEHAND);
    ZeroToPas(paramPtr, MYTIMEHAND^, Str);
    gl.y := StrToExt(paramPtr, Str);
    DisposHandle(MYTIMEHAND);

    MYTIMEHAND := GetGlobal(paramPtr, 'v');
    hlock(MYTIMEHAND);
    ZeroToPas(paramPtr, MYTIMEHAND^, Str);
    gl.v := StrToExt(paramPtr, Str);
    DisposHandle(MYTIMEHAND);

    MYTIMEHAND := GetGlobal(paramPtr, 'isPendown');
    hlock(MYTIMEHAND);
    ZeroToPas(paramPtr, MYTIMEHAND^, Str);
    gl.IsPendown := StrToBool(paramPtr, Str);
    DisposHandle(MYTIMEHAND);
end;

{ procedure SetGlobalValue (paramPtr:XcmdPtr;gl:global)          }
procedure setGlobalValue;
    var
        MYTIMEHAND: handle;
        Str: str255;
begin
    ExtToStr(paramPtr, gl.x, str);
    MYTIMEHAND := PasToZero(paramPtr, Str);
    hlock(MYTIMEHAND);
    SetGlobal(paramPtr, 'x', MYTIMEHAND);
    DisposHandle(MYTIMEHAND);

    ExtToStr(paramPtr, gl.y, str);
    MYTIMEHAND := PasToZero(paramPtr, Str);
    hlock(MYTIMEHAND);
    SetGlobal(paramPtr, 'y', MYTIMEHAND);
    DisposHandle(MYTIMEHAND);

    ExtToStr(paramPtr, gl.v, str);
    MYTIMEHAND := PasToZero(paramPtr, Str);
```

```
    hlock(MYTIMEHAND);
    SetGlobal(paramPtr, 'v', MYTIMEHAND);
    DisposHandle(MYTIMEHAND);

    BoolToStr(paramPtr, gl.isPendown, str);
    MYTIMEHAND := PasToZero(paramPtr, Str);
    hlock(MYTIMEHAND);
    SetGlobal(paramPtr, 'isPendown', MYTIMEHAND);
    DisposHandle(MYTIMEHAND);
  end;

end.
```

XCMDの一例として，ForwardのList-2を以下に示す．

List-2　forward.tlsp

```
{  FORWARD                                                              }
{                    XCMD 4003                                          }
{ USAGE:                                                                }
{ Forward Length                                                        }
{Project Build Order:                                                   }
{interface.lib                                                          }
{      THINKPascalのLibrary                                             }
{DRVRRuntime.lib                                                        }
{      THINKPascalのLibrary                                             }
{HyperXCmd.TLSP                                                         }
{      Apple社の提供するHyperTalk　のCallback Routines                  }
{XcmdGlue.TLSP                                                          }
{      Apple社の提供するGlue Rotines                                    }
{XCmdGlueUtils.TLSP                                                     }
{      LANGUAGE SYSTEMS社より提供されたGlue Rotinesを使うためのUtility  }
{      上のHyperXCmd.TLSP,XcmdGlue.TLSP,XCmdGlueUtils.TLSPは            }
{      LANGUAGESYSTEM社のWildThingsに含まれるもので，THINKPascalと共に  }
{      提供されるものとは異なる                                         }
{global.TSLP                                                            }
{FORWARD.TLSP                                                           }

unit DummyUnit;

interface
  uses
```

```
    HyperXCMD, XCMDGlue, XcmdGlueUtils, global;

  procedure ENTRYPOINT (paramPtr: XCmdPtr);
  procedure main (paramPtr: XCmdPtr);

implementation

  procedure FD (paramPtr: XCmdPtr);
  FORWARD;

  procedure ENTRYPOINT (paramPtr: XCmdPtr);
  begin
      FD(paramPtr);
  end;

{**************************************}
  procedure FD;
      var
          theString: Str255;
          startx, starty, endx, endy: Str255;
          startXp, startYp, endXp, endYp: LongInt;
          gl: global;
          x, y, length: Extended;
  begin
{ Check parameters     }
{   exit if parameter count is not 1   }
      if ((paramPtr^.paramCount < 1) or
          (paramPtr^.paramCount > 1))
          then
              begin
                  sysbeep(9);
                  theString := 'answer "FORWARD Length"';
                  SendCardMessage(paramPtr, theString);
                  EXIT(FD);
              end;
      length := RealArgument(paramPtr, 1);
      GetGlobalValue(paramPtr, gl);
      x := gl.x + length * sin(gl.v / 180 * pi);
      y := gl.y + length * cos(gl.v / 180 * pi);
      if gl.isPendown then
          begin
{ They are converted to integer so they can be mapped on screen.}
              startXp := round(CenterX + gl.x);
```

```
            startYp := round(CenterY - gl.y);
            endXp := round(CenterX + x);
            endYp := round(CenterY - y);
            NumToStr(paramPtr, startXp, startx);
            NumToStr(paramPtr, startYp, starty);
            NumToStr(paramPtr, endXp, endx);
            NumToStr(paramPtr, endYp, endy);
            SendCardMessage(paramPtr,
                        'choose line tool ');
            theString := CONCAT('drag from ', startx,
             ',', starty, ' to ', endx, ',', endy);
            SendCardMessage(paramPtr, theString);
        end;
{ Set globals }
    gl.x := x;
    gl.y := y;
    SetGlobalValue(paramPtr, gl);
  end;
{*************************************}
  procedure main;
  begin
    FD(paramPtr);
  end;

end.
```

このProgramをResourceとして，コンパイルしてResEdit(Resource Editor)を用いて，HyperCardのStack Resourceに張り付ける．これにより，外部Commandを使用できるようになる．

4. Program開発支援用Stack

HyperTurtleを使って図形を描くProgramを開発支援するためのStackを示す．

このStackは2種類のBackgroundを持っている．一つはProgramを書くためのもので(図4-1)これをProgramCard，もう一つは描いた絵を保存(図 4-2)するためのものであるがこれをPictureCardと名付けることにする．

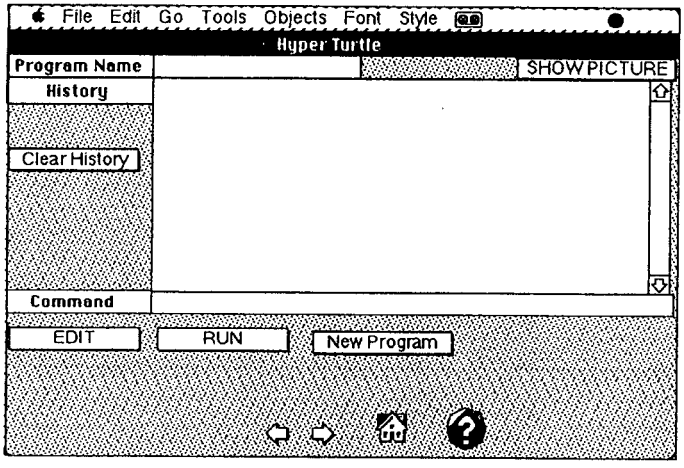


図4-1

Programを書くためのCardの主な機能(ProgramCard)

EditButtonをマウスでクリックすると，ScriptEditorが表示され，ここに，Scriptを記述する．CommandFieldに実行するCommandを書き込みカーソルがこのFieldにある時，ReturnKey，EnterKeyを押すか，RunButtonをマウスでクリックすると，別のCard（図4-2）が現われ，Commandが実行される．実行し終わると自動的にこのCardに戻るが．PictureButtonをマウスでクリックすると図4-2のCardが現われる．

一度実行したCommandはHistoryFieldに残され，クリックすることにより，その行の命令がCommandFieldに書き込まれる．

このようにProgramをパラメーターをいろいろ変えながら，実行し開発しやすいように設計した．また，このCardには，見えないが，これと組をなす絵を描くためのPictureCardのCardIDを保存するためのFieldがある．

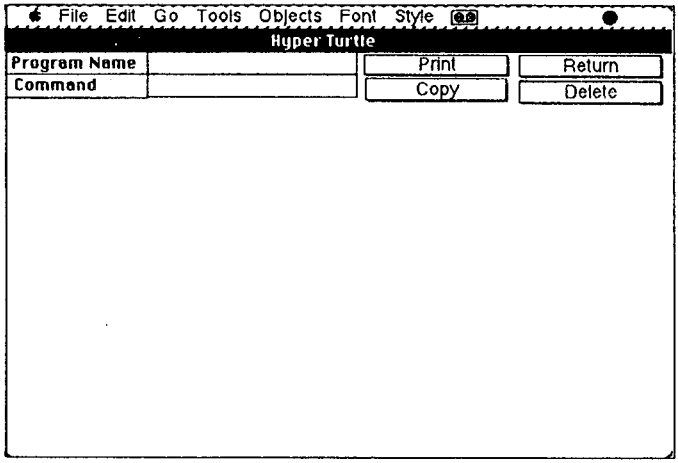


図4-2

描いた絵を保存するためのCardの主な機能 (PictureCard)

図6-1のCardのRunButtonをマウスでクリックするとこのCardが現われる．Program名，Command名は自動的にProgramCardからコピーされ，絵が描かれ元のCardに戻る．ProgramCardのPictureButtonをマウスでクリックすれば，このCardが再び現われる．この時，PrintButtonをマウスでクリックすればこのCardが印刷される．ReturnButtonをクリックするかReturnKeyを押すことにより，元のCardに戻る．CopyButtonをマウスでクリックすれば，実行結果をCopyして保存することが出来るし，DeleteButtonをマウスでクリックすれば，そのCardを消去出来る．

以下に，Stackに含まれるスクリプトを示す．

```
** STACK SCRIPT:*********************************************
on openStack
  global V,X,Y,isPendown,saveLevel
  put 0 into V
  put 0 into X
  put 0 into Y
  put true into isPendown
  put the userLevel into saveLevel
  if the userLevel < 5 then set userLevel to 5
end openStack
```

-> HyperCardは使用者のレベルをBrowsing, Typing, Painting, Authoring, Scriptingの5つに分けているが，それをScriptingのレベルにする．

```
on closeStack
  global savelevel
  set userLevel to saveLevel
end closeStack
```

-> 使用前のレベルに戻す．

```
on fd l
  forward l
end fd

on bk l
  back l
end bk

on rt a
  right a
end rt

on lt a
  left a
end lt

on cs
  clearscreen
end cs

on pu
  penup
end pu

on pd
  pendown
end pd
```

->fdからpdまではHyperTurtle命令での省略形が出来るようにした．

```
on Home
  SetXY 0,0
  SetHeading 90
```

```
end Home
->原点にもどり，真上を向く．
function xcor
  global x
  return x
end xcor
->現在のTurtleのx座標を示す関数
function ycor
  global y
  return y
end ycor
->現在のTurtleのx座標を示す関数

** BACKGROUND SCRIPT: ProgramCard ***************************
on Run
  get  line 1 of bkgnd field "Command"
  put it into L
  put it&return after bkgnd field "History"
  get  line 1 of bkgnd field "ProgramName"
  put it into ProgramName
  put the id of this card into ProgramID
  get bkgnd field "PictureId"
  if it is empty then
    push this card
    go to last card of bkgnd "PictureCard"
    domenu New Card
    get id of this card
    pop card
    put it into bkgnd field "PictureId"
  end if
  visual effect zoom open
  go to it
  put ProgramId into bkgnd field "ProgramID"
  put ProgramName into bkgnd field "ProgramName"
  put L into bkgnd field "Command"
  send L to  recent card
  choose browse tool
  go to ProgramId
end Run
```

->CommandをHistoryFieldに付け加え，組になるPictureCardがまだ出来ていなかったらつくり，PictureCardに移りCommandを実行し再びProgramCardに戻る．

```
** BACKGROUND Button SCRIPTS  ******************************
** Button: bkgnd Button "EDIT"  ****************************
on mouseUp
  edit script of this card
end mouseUp
```
-> ScriptEditorを呼び出す．

```
** Button: bkgnd Button "SHOWPICTURE"***********************
on mouseUp
  put the id of this card into ProgramID
  get bkgnd Field "PictureId"
  if it is empty then
    push this card
    go to last card of bkgnd "PictureCard"
    domenu New Card
    get id of this card
    pop card
    put it into bkgnd Field "PictureId"
  end if
  visual effect zoom open
  go to it
  put ProgramId into bkgnd Field "ProgramID"
end mouseUp
```
->組になっているPictureCardに移る，まだ命令を実行していない時は，組になるPictureCardをつくりそれに移る．

```
** Button: bkgnd Button "Home"  ******************************
on mouseUp
  visual effect iris close
  go home
end mouseUp
```
->HomeCardに移る．

```
** Button: bkgnd Button "Previous" ************************
on mouseUp
  visual effect wipe down
```

```
  go to previous card
end mouseUp
```
->前のCardに移る.
```
** Button: bkgnd Button "Next" ******************************
on mouseUp
  visual effect wipe up
  go to next card
end mouseUp
```
->次のCardに移る.

```
** Button: bkgnd Button "Help" ******************************
on mouseUp
  help
end mouseUp
```
->HelpCardに移る. (将来ここに簡単なHyperTurtleの使用法を書き込むつもりである.)

```
** Button: bkgnd Button "Clear History" ********************
on mouseUp
  put "" into bkgnd Field "History"
end mouseUp
```
->HistoryFieldの消去

```
** Button: bkgnd Button "RUN" ******************************
on mouseUp
  Run
end mouseUp
```
->Run Buttonを押したときBackgroundのRUNを実行する.

```
** Button: bkgnd Button "New Program"  **********************
on mouseUp
  doMenu "New Card"
end mouseUp
```
-> 新しいProgramCardを作る.

```
** BACKGROUND Field SCRIPTS  ********************************
** Field: bkgnd Field "HISTORY"******************************
on mouseUp
  select the clickLine
```

```
  put the selection into bkgnd Field "COMMAND"
end mouseUp
```
-> History Fieldをでクリックした行をCommandFieldにコピーする.
```
** Field: bkgnd Field "COMMAND" ****************************
on ReturnInField
  Run
end ReturnInField
```
-> commandFieldでRcturnKeyを押されたとき,BackgroundのRUNを実行する.

```
on enterInField
  Run
  pass enterInField
end enterInField
```
-> commandFieldでReturnKeyを押されたとき,BackgroundのRUNを実行する.

```
** BACKGROUND SCRIPT: PictureCard ***************************
on returnKey
  get bkgnd Field "programId"
  go to  it
end returnKey
```
-> PictureCard上でRerurnKeyが押されたら組になるProgramCardに戻る.

```
** BACKGROUND Button SCRIPTS ********************************
** Button: bkgnd Button "Print" ****************************
on mouseUp
  doMenu "Print Card"
end mouseUp
```
-> 画面のコピーをprinterに出力する.

```
** Button: bkgnd Button "Return" ***************************
on mouseUp
  get line 1 of bkgnd Field "ProgramID"
  go to it
end mouseUp
```
-> 組になるProgramCardに戻る.

```
** BUTTON: bkgnd button "Copy" *****************************
on mouseUp
  doMenu "Copy Card"
```

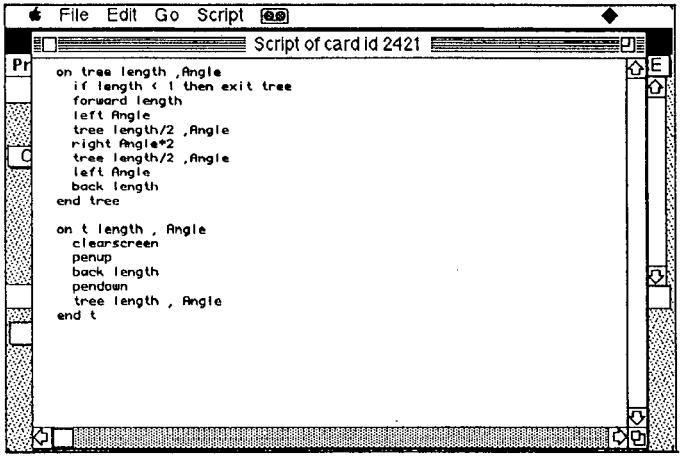
```
  doMenu "Paste Card"
end mouseUp
```
->CardのCopyをとる.

```
** BUTTON: bkgnd button "Delete"  ***************************
on mouseUp
  put the id of this card into PictureID
  get line 1 of bkgnd field "ProgramID"
  go to it
  if PictureID is  line 1 of  field "PictureID" then
    put "" into field "pictureID"
  end if
  go to PictureId
  doMenu "Delete Card"
end mouseUp
```
-> Cardを消去する，もしProgramCardと組になるCardならば，ProgramCardに記録してある，Card IDも消去する.

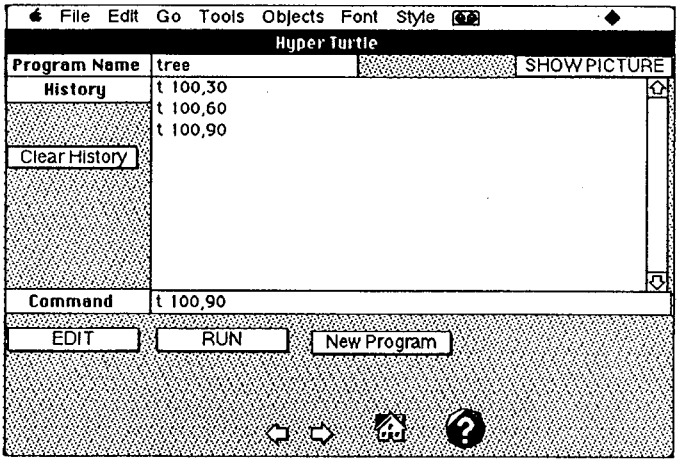
使用例

Button"Edit"押してScriptEditorを開いたところ.

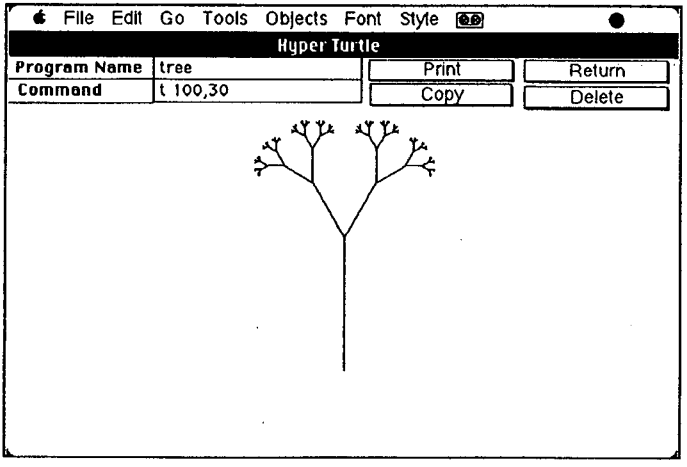
Scriptditorは自動的にScriptの構文に合わせてIndentされるので大変便利である.



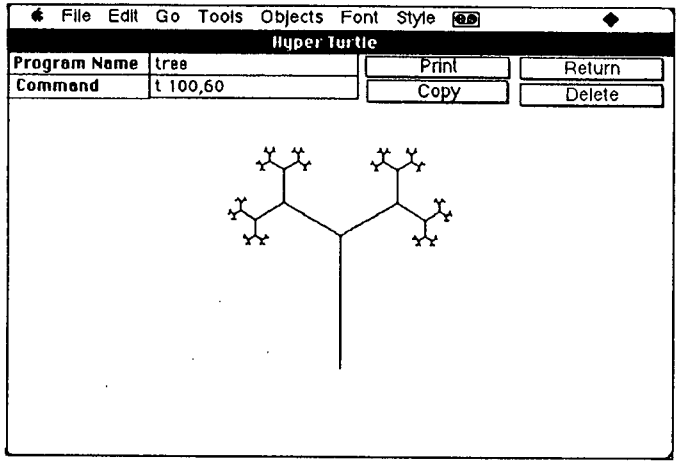
実行されたCommandはこのようにHistoryFieldに保存される.



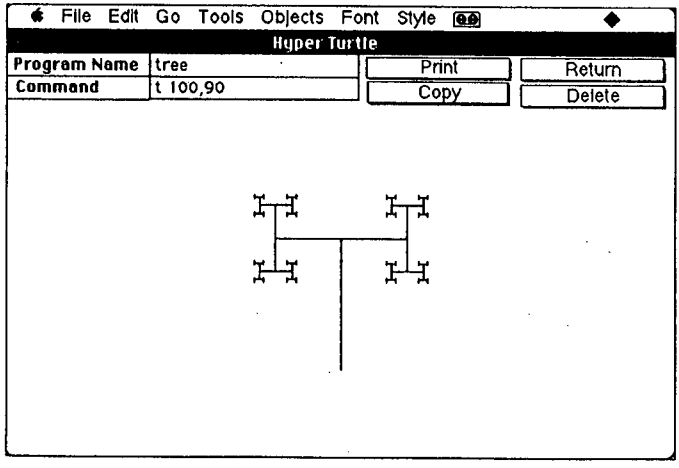
PictureCardのButton"Copy"を押すことにより，同じProgramCardで作られた図形を複数保存出来る．



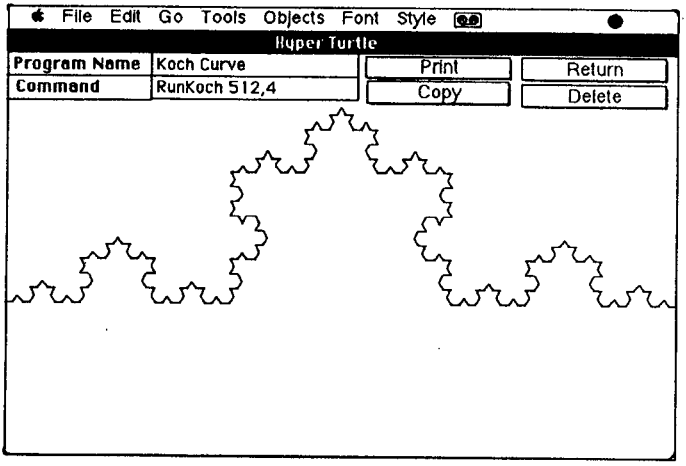
t 100,30の実行結果



t 100,60の実行結果

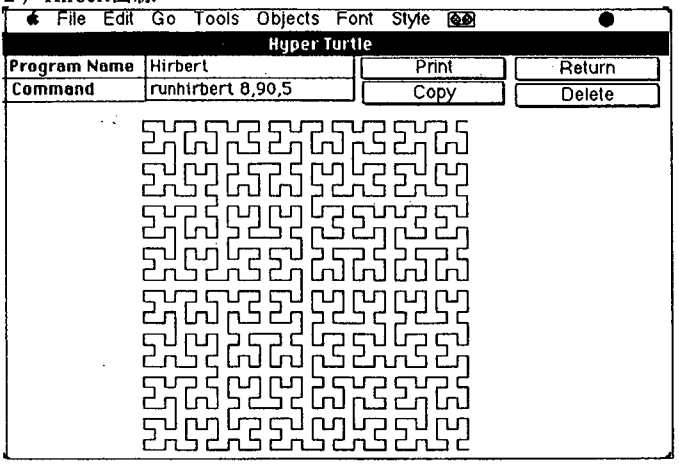


実行例（1）Koch曲線




```
on Koch l,n
  if n > 0 then
    Koch l/3 , n-1
    left 60
    Koch l/3 , n-1
    right 120
    Koch l/3 , n-1
    left 60
    Koch l/3 , n-1
  else
    forward l
  end if
end koch
on runkoch l,n
  clearscreen
  penup
  setxy -256,-50
  right 90
  pendown
  Koch l,n
end runkoch
```

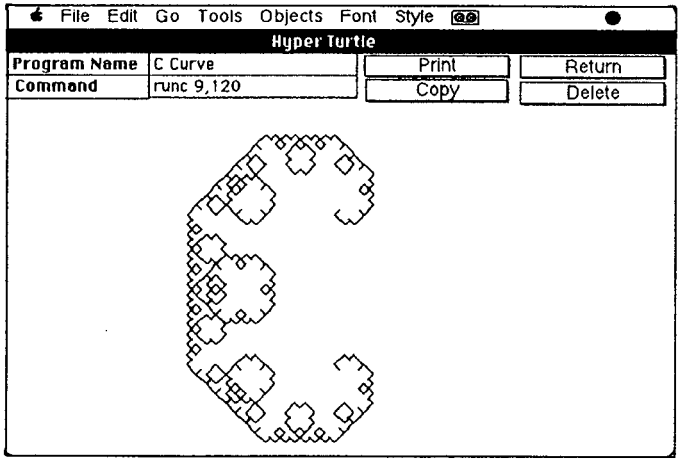
実行例（2）Hirbert曲線




```
on hirbert size,angle,n
  if n=0 then exit hirbert
  Left angle
  hirbert size ,-angle,n-1
  Forward size
  Right angle
  hirbert size, angle, n-1
  Forward size
  Hirbert size, angle, n-1
  Right angle
  Forward size
  hirbert size ,-angle, n-1
  Left angle
end hirbert

on runHirbert size,angle,n
  PenUp
  setxy 100, -160
  PenDown
  hirbert size,angle,n
end runHirbert
```

実行例（3） C曲線




```
on c n,l
  if n = 0 then
    fd l
    exit c
  end if
  lt 45
  c n-1, l/sqrt(2)
  rt 90
  c n-1, l/sqrt(2)
  lt 45
end c

on runc n,l
  cs
  pu
  sety -100
  pd
  c n,l
end runc
```

6. おわりに

CやPascalだけで，今回作った「TurtleGraphicsのProgarmを開発・実行・結果の保存」をすべて行なうようなアプリケーションをつくることは非常に難しい．

（InsideMacintoshという分厚い何冊もの本を読み，Macintosh のシステムルーチンについて，精通しなければとても書けない．）

HyperCardでPrgramの主要なインターフェイスを組立，HyperCardではどうしても実現できない機能・処理速度に問題があるところをXCMD,XFCNで補う今回の方法は，Program の専門家でないものが，Macintoshのアプリケーションをつくるのに適した方法である．(同じCやPascalで書くにしても，XCMD,XFCNは一つの形式を覚えれば比較的簡単に作ることができる)

また，一般にProgramはTextFileとして，実行結果は印刷するかあるいはPaintFileとして保存されるが，今回の方法ではProgramの実行結果もCard に記録される．

これはProgramを見直すとき，あるいはParameterをいろいろ変えて実行結果を比べるときなどに非常に有効だった．

また，視覚的な側面からProgram を教えられるので，Progarmの教育にも用いて行きたいと思う．その際，HyperCardのHyperTextの機能をいかして，Tutor用ののStackなども作りたい．

現在HyperCardを用いて，数学学習環境・支援システムの開発中であるが，HyperTurtleをグラフツールなどに発展させるつもりである．

今回は図形を描かせるだけであったが，簡単な数値計算用(数値積分・級数の和)のProgramを開発し，その記録を残すStackなども作ってみたいと思っている．

謝辞　本研究は平成３年度安倍能成記念教育基金学術助成金の援助のもとに行われた．記して謝意を表する．

参考文献

Gary Bond: XCMD's for HyperCard,Management Information Source Inc.,1988

Danny Goodman: HyperCard Developer's Guide,A Bantam Book: 1988

梅村恭司: 考える道具としてのMacintosh/HyperCard,共立,1989

大重美幸: HyperTalkハンドブック,BNN,1988

Karen R.,Guy M,Deninis M. and Rich Norling : Wild Things Reference Manual,LANGUAGE SYSTEMS Corp.,1988

西林瑞夫:OSシリーズ6　Macintosh[増補版],共立,1988

Harold Abelson:LOGO －エーベルソンのLogo入門－五藤博義訳,啓学出版,1985